APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: GRID LANDSCAPE COMPONENT

APPLICANT: EROL BOZAK AND ALEXANDER GEBHART

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV331001525US

November 12, 2003
Date of Deposit

GRID LANDSCAPE COMPONENT

CROSS-REFERENCE TO RELATED APPLICATIONS

This application incorporates by reference the content of U.S. Provisional Application No. 60/490,818, Express Mail Number, EV 331001684 US, filed July 28, 2003, to Erol Bozak *et al.*, entitled GRID COMPUTING MANAGEMENT.

TECHNICAL FIELD

The present invention relates to data processing by digital computer, and more particularly to a grid landscape user interface.

BACKGROUND

In today's data centers, the clusters of servers in a client-server network that run business applications often do a poor job of managing unpredictable workloads. One server may sit idle, while another is constrained. This leads to a "Catch-22" where companies, needing to avoid network bottlenecks and safeguard connectivity with customers, business partners and employees, often plan for the highest spikes in workload demand, then watch as those surplus servers operate well under capacity most of the time.

In grid computing, all of the disparate computers and systems in an organization or among organizations become one large, integrated computing system. That single integrated system can then handle problems and processes too large and intensive for any single computer to easily handle in an efficient manner.

More specifically, grid computing is a form of distributed system wherein computing resources are shared across networks. Grid computing enables the selection, aggregation, and sharing of information resources resident in multiple administrative domains and across geographic areas. These information resources are shared, for example, based upon their availability, capability, and cost, as well as a user's quality of service (QoS) requirements. Grid computing can mean reduced cost of ownership, aggregated and improved efficiency of computing, data, and storage resources, and enablement of virtual organizations for applications and data sharing.

SUMMARY

In one aspect, the invention features a graphical user interface (GUI) including a structure with columns and rows, each of the rows representing services in a grid computing network, the rows structured hierarchically with respect to an application where a service belongs, a type of service and concrete service instances.

Embodiments may include the following. In some cases, each service instance row is associated to a place in the grid-like structure representing where it is instantiated. In these cases, the columns may represent grid nodes.

In another aspect, the invention features a graphical user interface (GUI) describing a set of services managing a portion of a computer grid, the GUI including a matrix-like structure with columns and rows, each column representing a computer from a set of computers in the computer grid, each computer from the set of computers having a grid manager, and each row representing a grid manager or other application service, positions of labels in the structure indicating which computer currently runs which grid manager or other application service. The GUI also includes a column representing a first computer from the set of computers running a first grid manager, and one or more columns representing one or more computers from the set of computers running one or more grid managers having an inferior relation with the first grid manager.

Embodiments may include one or more of the following. The GUI further includes a shrinkable structure that hides the labels representing grid managers or other application services in the matrix-like structure. The rows representing application services are structured by application class.

In another aspect, the invention features a method including receiving a request to view a sub grid network of a grid network, the sub grid network representing a root node and nodes with inferior relations to the root node, the nodes representing grid managers managing one or more services running on computers in the grid network. The method also includes querying a grid manager representing the root node for its status and addresses of nodes with inferior relations, querying inferior grid managers for current status, and displaying a state of the root and inferior grid managers and for each grid manager, a computer system running the grid manager.

These and other embodiments may have one or more of the following advantages. Using the GUI, a user is enabled to visualize the landscape of a grid computing environment.

The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram of a grid computing environment.

FIG. 2 is a flow diagram for discovering and reserving resources in the grid computing environment of FIG. 1.

FIG. 3 is a flow diagram for installing, running, and removing applications in the grid computing environment of FIG. 1.

FIG. 4 is a block diagram of a computer device in the grid computing environment of FIG. 1.

FIG. 4A is a flow diagram for starting up an application in the computer device of FIG. 4.

FIG. 5 is a flow diagram for starting up grid managers in the grid computing environment of FIG. 1.

FIG. 5A is a block diagram of the grid computing environment of FIG. 1 that is augmented with another computer device.

FIG. 6 is a block diagram of an exemplary a grid graphical user interface (GUI) component for visualization of a grid computing environment.

FIG. 7 is a block diagram of a grid browser component.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

As shown in FIG. 1, services in a grid computing environment **100** manage computational resources for applications. The grid computing environment **100** is a set of distributed computing resources that can individually be assigned to perform computing or data retrieval tasks for the applications. The computational resources include computer devices **12**, **14**, **16**, **18**, **20**, and **22**. The computer devices communicate using a network **8.** The applications have scalable computational requirements. For example, an example application that uses computer devices **12, 14, 16**, **18**, **20**, and **22** in the grid computing environment **100** is an internet pricing configurator. The computer device **12** provides network access to pricing information to users via web browsers on computer devices that are connected to the internet. The web

browsers can be any application able to display content and/or execute applications such as web pages, media files, and programs, such as Netscape Navigator®, Microsoft Internet Explorer®, and similar applications.

In this example, a web server on computer device **12** provides pricing information to the users. Calculation parameters for each price to be calculated are passed by an IPC dispatcher **116** to IPC servers **120**, **122**, **124**, and **126** that execute on computer devices **12**, **14**, **16**, and **18**, respectively. Due to the flexibility of the web server and applications on the internet, the number of users can vary. This generates dynamic computational requirements for the internet pricing configurator. An IPC manager **118** communicates with services in the grid computing environment **100** so that the services can allocate and deallocate computational resources (e.g., processors in computer devices **12**, **14**, **16**, **18**, **20**, **22**) based on the dynamic computational requirements of the internet pricing configurator. Allocating and deallocating computational resources in this manner allows computer devices **12**, **14**, **16**, **18**, **20**, or **22** to be designated as general-purpose computational resources and not solely dedicated to handling peak demands of the internet pricing configurator application. The IPC manager **118** coordinates with the IPC dispatcher **116** so that the IPC dispatcher **116** has access to resources in network **8**.

This capability to allocate and deallocate the resources in the grid computing environment **100** enables the IPC manager **118** to locate and use available computational resources on an "as needed" basis. Once resources are located, the IPC manager **118** can use services in the grid computing environment **100** to install the IPC servers **120**, **122**, **124**, and **126** as applications on computer devices in the grid computing environment **100**. The IPC dispatcher **116** uses Web Service Definition Language (WSDL) interfaces defined in the Open Grid Services Infrastructure (OGSI) Version 1.0 by Tuecke *et al* to manage and exchange the information flow between the IPC dispatcher **116** and IPC servers **120**, **122**, **124**, and **126**. For example, the OGSI WSDL interfaces can be used to pass computation parameters for pricing calculations from the IPC dispatcher **116** and the IPC servers **120**, **122**, **124**, and **126**. The OGSI WSDL interfaces can also be used to pass completed results from the IPC servers **120**, **122**, **124**, and **126** back to IPC dispatcher **116**. The OGSI Version 1.0 is incorporated herein by reference. The OGSI WSDL interfaces enable the controlled, fault-resilient, and secure management of the grid computing environment **100** and applications such as the internet pricing configurator.

While the IPC dispatcher **116** uses IPC servers **120**, **122**, **124**, and **126** to perform calculations for users, services in the grid computing environment **100** monitor resource utilization on computer devices in the grid computing environment **100** running the IPC servers **120**, **122**, **124**, and **126**. The services also send this utilization information to the IPC manager **118**. Based on a comparison between utilization requirements and current resource loading, the IPC manager **118** can dynamically inform services in the grid computing environment **100** to allocate more resources for IPC servers **120**, **122**, **124**, and **126** or deallocate resources to keep utilization of resources in the grid computing environment **100** at a desired level.

Grid managers **152**, **154**, **156**, **160**, **162**, and **164** are resident in computer devices **12**, **14**, **16**, **18**, **20**, and **22**, respectively. Within the grid computing environment **100**, pairs of grid managers can have directional relations that classify one grid manager as superior to another grid manager. A grid manager can have more than one superior relations with other grid managers. For example, grid manager **152** has a superior relation with grid managers **154** and **156**. A grid manager can also have more than one inferior relations with other grid managers. Through these hierarchical relations, IPC manager **118** does not need access to a list of all computer devices in network **8** to use the computational resources in the grid computing environment **100**. IPC manager **118** is only required to have access to a network address of one computer device running a grid manager (e.g., computer device **12** running grid manager **152**) and this grid manager uses its relations with other grid managers running on other computer devices to provide IPC dispatcher **116** with indirect access to other computer devices in the grid computing environment **100**.

A grid manager (e.g., **152**, **154**, **156**, **160**, **162**, and **164**) maintains a first list of all superior relations with other grid managers and a second list of all inferior relations with other grid managers. Each grid manager maintains an "always open" communications channel to all the grid managers in these lists over network **8** using, for example, the aforementioned OGSI WSDL interfaces on transmission control protocol (TCP), hypertext transfer protocol (HTTP), and simple object access protocol (SOAP). These lists and corresponding communication channels can be modified, allowing a dynamic reconfiguration of the grid hierarchy during runtime. This also allows a failing grid manager to be dynamically replaced in the hierarchy. For example, referring to FIG. 1, if grid manager **154** fails, then grid manager **152** loses its connection to grid managers **160** and **162**. In this case, relations between grid managers can be

modified so that grid manager **152** has new superior relations to grid managers **160** and **162**. Likewise, grid managers **160** and **162** have new inferior relations to grid manager **152**.

As shown in FIG. 2, an application start process **200** is designed so applications (e.g., internet pricing configurator) get necessary resources allocated in the network **8** before executing on a computer device (e.g., **12**, **14**, **16**, **18**, **20**, or **22**). Process **200** also guarantees if similar applications are trying to start at the same time on the same resource on a computer device that the two or more applications do not collide or interfere with each other. For example, the IPC manager **118** can require that an IPC server (e.g., **120**) be the only application executing on a processor in computer device **14** for quality of service (QoS). In this case, another application would interfere if the other application simultaneously attempted to execute on the processor in computer device **14**.

Process **200** includes IPC manager **118** (or some other application) sending **(202)** requirements for computational resources to query a grid manager (e.g., **154**) to determine if there are resources matching these requirements available in the grid computing environment **100**. These requirements specify information pertaining to resources in a computer device such as required number of processors, required percentage of utilization for those processors, main memory, and network speed. The query can also include information to which hierarchy level (in the grid computing environment **100**) the query should be propagated. Process **200** includes grid manager **154** receiving **(204)** the requirements.

To respond to the query for available resources from IPC manager **118**, process **200** includes grid manager **154** matching **(206)** the requirements against resources known to grid manager **154**. These resources include resources (e.g., processor **40**) in computer device **14** that are directly managed by grid manager **154**. Resources directly managed by grid manager **154** that are currently available and meet the requirements are added to a resource-query list maintained by grid manager **154**.

Grid manager **154** also sends the query to grid managers **160** and **162** having inferior relations with grid manager **154**. Process **200** includes grid managers **160** and **162** responding **(208)** to the query by sending to grid manager **154** lists of resources (e.g., processors on computer devices **18, 20**) that meet the requested requirements and are available and known to grid managers **160** and **162**, respectively. These resource-query lists of resources that are known to grid managers **160** and **162** can also include resources managed by grid managers (not shown)

with inferior relations to grid managers **160** and **162**. Grid manager **154** adds these resource-query lists of available resources from grid managers **160** and **162** to its resource-query list of available resources meeting the requested requirements. If process **200** determines (**210**) that there is at least one resource (e.g., processor **40**) in this resource-query list, then grid manager **154** sends (**214**) this resource-query list to IPC manager **118**. Otherwise, if process **200** determines (**212**) that grid manager **154** has a relation with a superior grid manager (e.g., grid manager **152**), grid manager **154** sends (**202**) the query for available resources to grid manager **152**. In response to this query, grid manager **152** does not send a redundant query back to grid manager **154** having an inferior relation with grid manager **152**.

Process **200** includes grid manager **154** sending (**214**) the list of available resources along with addresses of their corresponding grid managers in the network **8** that match the requirements. The IPC manager **118** selects a resource (e.g., on computer device **16**) from the list and requests (**216**) a reservation of the resource on computer device **16** to the grid manager **154** managing the resource on computer device **16**. If the resource in computer device **16** is still available for reservation (**218**) and the reservation succeeds, grid manager **154** sends (**220**) a reservation number to the IPC manager **118**. This reservation means that the IPC manager **118** is guaranteed and allocated the requested resource on the computer device **16** in the grid computing environment **100**. The grid manager **154** handles queries for available resources from applications such as IPC manager **118** using independent processing threads of execution. Thus, the grid manager **154** uses a semaphore to ensure that the same resource (e.g., processor **40**) is not assigned multiple reservation numbers for different applications simultaneously requesting the same resource.

If the grid manager determines that the requested resource in computer device **16** is not available for reservation and the reservation fails, the IPC manager **118** selects the next available resource in the list and requests (**216**) the reservation of this next available resource. If the IPC manager **118** receives a registration number and a timeout measured from the sending of the registration number does not expire (**222**), the IPC manager **118** starts (**224**) the IPC server **122** on the processor **40** resource in computer device **16**. Starting the IPC server **122** is initiated by passing the reservation number and an application file to the grid manager **156** and then grid manager **156** reads the application file to install and execute the IPC server **122** on computer device **16**.

As shown in FIG. 3, process **250** installs an application (e.g., IPC server **122**) on a computer device (e.g., **14**) in the grid computing environment **100** to set up an available resource for the application, using the available resource, and removing or deinstalling the application to free up the resource for use by subsequent applications when the resource is no longer needed. Process **250** includes IPC manager **118** transferring (**252**) an application file containing code for IPC server **122** in addition to instructions on how to install, customize, track and remove the application from computer device **14** so that the grid manager **154** can return computer device **14** to an original state after executing the application.

IPC manager **118** transfers the application file using a file transfer protocol (FTP), hypertext transfer protocol (HTTP), or a file copy from a network attached storage (NAS) for example, to computer device **14** as a single file, such as a compressed zip file. Within this zip file there is information about installing and customizing the application IPC server **122**. This information is represented by a small executable program or extended markup language (XML) document that is extracted and interpreted (**254**) by an installation and customizing engine (not shown) in grid manager **154**. Process **250** includes grid manager **154** installing (**256**) and running (**258**) the application. During installation (**256**), customization and execution (**258**) of the application, all changes to the computer device **14** are logged so that when the application is terminated (**260**) or deinstalled by grid manager **154** upon request by IPC manager **118**, grid manager **154** removes the application from the computer device **14** and also removes (**262**) any other changes to computer device **14** that were done when installing and running the application. Thus, the computer device **14** reverts to its original state prior to execution of the application and all of the resources of computer device **14** are again available for use by a subsequent application. This allows the resources to become available after running the application without rebooting computer device **14**. These changes include space in memory (e.g., **32**) allocated to store and run application code in addition to other changes such as allocation of communication ports.

In some examples, multiple applications can simultaneously run on resources in a single computer device (e.g., **14**). Applications for the grid computing environment **100** are classified in part based on their resource requirements. Some changes to a computer device to run an application are only required for the first execution of an application of its class and subsequent executions do not require these changes. In these examples, grid manager **154** only does the

changes for the first execution. Furthermore, when deinstalling the applications, grid manager **154** only removes the changes for the last application that was executed and terminated.

After installing applications on computer devices in the grid computing environment **100**, grid managers are configured to start or stop the processes of these applications upon request. In the example of the internet pricing configurator (IPC) application, grid manager **154** is configured to start or stop IPC server **122** on computer device **14** after installing IPC server **122** on computer device **14**. The IPC manager **118** requests grid managers to start or stop IPC servers in the grid computing environment **100** based on current utilization of resources in the grid computing environment **100**. After stopping IPC server **122** on computer device **14**, IPC manager **118** waits a prespecified amount of time and then requests grid manager **154** to deinstall IPC server **122** if current resource utilization does not indicate a need to start IPC server **122** again. Furthermore, as mentioned previously, grid managers monitor resource utilization on computer devices such as computer device **14** running applications (e.g. IPC servers **120**, **122**, **124**, and **126**) and send this utilization information to IPC manager **118**.

In many examples, control of application processes on resources in a computer device is specific to the operating system (OS). The grid computing environment **100** is configured to handle different operating systems on computer devices. Furthermore, grid computing environment **100** is designed to handle different applications (e.g., internet pricing configurator) that do not have to be redesigned to execute on the grid computing environment **100**. A grid manager controls an application process in a general manner that decreases interdependence between development of grid manager code and application code. An interface is provided to application code to enable grid managers to discover, control (e.g., start, stop, halt, resume) and inspect or monitor a state of application processes. The interface is provided for operating system processes that are exposed by the operating system or hosting environment and includes three aspects. One aspect of the interface is process data, such as process identification, states, degree of resource consumption (such as Central Processing Unit (CPU), memory, socket bindings, or other resources that an application can use), and application specific data defined by a process data scheme.

A second aspect of the interface is managing operations, such as start, stop, wait, resume, change priority, and other operations defined by supported managing operations.

A third aspect of the interface is control bindings and definitions, such as process data scheme, supported managing operations, and communication bindings. Since not all applications running in the grid computing environment **100** have access to the same information and capabilities in these three aspects, the applications provide to grid managers a list of queries and commands that each application supports.

The interface provided to application code is an Application Program Interface (API). The API is a set of methods (embedded in software code) prescribed by the grid manager software by which a programmer writing an application program (e.g., internet pricing configurator) can handle requests from the grid manager.

As shown in FIG. 4, IPC server **122** includes an API **302** and a document **304**. Since the API **302** is adapted to different types of applications, the document **304** describes how grid manager **154** communicates with the IPC server **122** and what requests through the API **302** are supported by the IPC server **122**. Grid manager **154** reads document **304** before starting up IPC server **122**. In some examples, document **304** is written in XML and includes a Document Type Description (DTD) **306**. A DTD is a specific definition that follows the rules of the Standard Generalized Markup Language (SGML). A DTD is a specification that accompanies a document and identifies what the markups are that separate paragraphs, identify topic headings, and how each markup is to be processed. By including the DTD **306** with document **304**, grid manager **154** having a DTD "reader" (or "SGML compiler") is able to process the document **304** and can correctly interpret many different kinds of documents **304** that use a range of different markup codes and related meanings.

As shown in FIG. 4A, grid manager **154** uses process **350** to install applications such as IPC server **122**. Grid manager **154** reads (**352**) DTD **306** in document **304** to identify markups in document **304**. Grid manager **154** reads (**354**) document **304** using markups to identify communication parameters for communicating with IPC server **122**. Grid manager **154** sets up (**356**) communications with IPC server **122** based on the specifications of the communication parameters. Grid manager **154** communicates (**358**) with IPC server **122** using the communication parameters to send requests such as “Start”, “Stop”, and “Are you idle?”.

Before any applications (e.g., internet pricing configurator) can be executed on network **8**, grid managers **152**, **154**, **156**, **160**, **162**, and **164** are asynchronously started up on computer devices **12**, **14**, **16**, **18**, **20**, and **22**, and relations to other grid managers are established. As

shown in FIG. 5, process **400** initializes relations among grid managers. For each grid manager (e.g., grid manager **154**), the grid manager **154** starts up on computer device **14** by reading **(402)** a properties file. The properties file contains a list of addresses of computer devices with grid managers having superior relations to grid manager **154**. This list was described earlier as a first list of all superior relations with other grid managers. If **(404)** a superior grid manager (e.g., grid manager **152**) is specified in this list of addresses, grid manager **154** requests **(406)** to open a communication channel to the superior grid manager (e.g., **152**). If grid manager **152** is already started, then grid manager **152** responds by accepting the request of the opening of the communication channel from grid manager **152**. Process **400** includes grid manager **154** detecting **(408)** any requests for communication channels from grid managers (e.g., grid managers **160**, **162**) identified as having inferior relations with grid manager **154**. If process 400 determines **(410)** that there are some requests, grid manager **154** allows communication channels from the inferior grid managers (e.g., **160**, **162**). Process **400** includes grid manager **154** checking **(414)** if there are any pending requests for communication to grid managers having superior relations. If there are any pending requests, grid manager **154** requests **(406)** communication channels to grid managers. These communication channels are used for resource queries between grid managers (as described previously) and "heart beat" messages between grid managers to ensure that each grid manager in the grid computing environment **100** is functioning.

Once grid managers **152**, **154**, **156**, **160**, **162**, and **164** are running with established relations, the grid managers are used for the proper operation of the grid computing environment **100**. Often during the lifecycle of the grid computing environment **100** the functionality of the grid managers are enhanced. It is often not possible or convenient to shut down the grid computing environment **100** and start the grid computing environment **100** up with the enhancements. Grid managers **152**, **154**, **156**, **160**, **162**, and **164** are configured so that there is only a minimal impact on users of the grid computing environment **100** when a change happens. To enable this transparency, an API is provided for user interfaces to enable an administrator of grid computing environment **100** to access each of the grid managers **152**, **154**, **156**, **160**, **162**, and **164** individually or all together. The API is static in that it includes only one method, i.e., a string that contains a command typed by the administrator. The API is dynamic because the string can contain many different commands.

In some cases, the grid managers are developed using the Java programming language. In these cases, new commands issued to the grid managers can be supported by loading new or revised Java classes dynamically via classloaders. This dynamic access to code can be done without shutting down grid managers in the grid computing environment **100**. Using Java classloaders, each time an instance of a class for a grid manager is generated, the definition and behavior of the class can be updated to provide new functionality to the grid computing environment **100**.

Another way to modify the functionality of the grid computing environment **100** dynamically without shutting down the grid computing environment **100** is to change the hierarchical relations between grid managers, remove grid managers, or add new grid managers. The API provided for administration of the grid computing environment **100** is also configured to send strings to individual grid managers with commands to delete existing relations or add new relations.

For administrators of grid computing environment **100**, it is useful to visualize the applications and a grid manager on one computer device in the grid computing environment **100** as well as other computer devices running part of the grid management hierarchy in the form of grid managers with one or more levels of inferior relations to the grid manager. The view of these computer devices is referred to as a grid landscape. As shown in FIG. 6, a grid graphical user interface (GUI) **500** for visualization of a grid landscape, such as the grid computing environment **100**, includes GUI-elements visualizing an organization of services running on computer devices. The GUI **500** provides a grid-like structure with columns and rows. Rows represent services, which in turn are structured hierarchically with respect to the application where a service belongs to, the type of the service, and the specific service instances. Each service instance row is associated with a place in the grid computing environment **100** representing where it is instantiated. In this context, columns represent the computer devices in the grid landscape. Specifically, GUI **500** has three columns representing three computer devices **12**, **14**, and **16**. GUI **500** shows that grid manager **152** runs on computer device **12** with inferior grid managers **154** and **156** running on computer devices **14** and **16**, respectively. GUI **500** also shows internet pricing configurator services running on computer device **12**. These internet pricing configurator services include IPC dispatcher **116**, IPC server **120**, and IPC manager **118**.

The GUI **500** is dynamically refreshed with feedback from the grid managers and internet pricing configurator (or other application) services so that new services appear in GUI **500** to an administrator. Similarly, services that are shut down are removed in GUI **500**.

As shown in FIG. 7, a grid browser component **600** is a composite graphical user interface (GUI) for browsing grid managers on computer devices in the grid computing environment **100**. The component **600** displays a graph with curved edges and vertices. Vertices represent computer devices in the grid computing environment **100** and curved edges represent the directional association of grid managers on two computer devices (vertices) in the grid computing environment **100**. This association is hierarchical (i.e., superior/inferior). Each vertex displays the network address of a computer device as well as applications currently running on the computer device. For example, component **600** shows computer devices **12**, **14**, **16**, **18**, **20**, and **22** with IPC servers **118**, **120**, **122**, and **124**. In other examples (not shown), the grid browser component **600** shows non-hierarchical, peer to peer associations of grid managers with non-directional edges representing the associations.

The grid browser component **600** is context sensitive. Depending on the relationship among the grid managers on the computer devices (e.g., superior/inferior), computer devices are traversed in respect to a user's browsing history.

By clicking on a vertex representing a computer device in GUI **600** (e.g., computer device **14**), a user can automatically view a grid manager and applications running on the computer device and grid managers having inferior relations to the grid manager using GUI **500**. The user can pick a computer device and see relations between its grid manager and other grid managers. This connection between GUIs **500** and **600** is done using software that generates GUIs **500** and **600**.

The network **8** can be implemented in a variety of ways. The network **8** includes any kind and any combination of networks such as an Internet, a local area network (LAN) or other local network, a private network, a public network, a plain old telephone system (POTS), or other similar wired or wireless networks. Communications through the network **8** may be secured with a mechanism such as encryption, a security protocol, or other type of similar mechanism. Communications through the network **8** can include any kind and any combination of communication links such as modem links, Ethernet links, cables, point-to-point links,

infrared connections, fiber optic links, wireless links, cellular links, Bluetooth®, satellite links, and other similar links.

The network **8** is simplified for ease of explanation. The network **8** can include more or fewer additional elements such as networks, communication links, proxy servers, firewalls or other security mechanisms, Internet Service Providers (ISPs), gatekeepers, gateways, switches, routers, hubs, client terminals, and other elements.

Computer devices **12**, **14**, **16**, **18**, **20**, and **22** communicate over medium **10** using one of many different networking protocols. For instance, one protocol is Transmission Control Protocol/Internet Protocol (TCP/IP) combined with SOAP (Simple Object Access Protocol).

Embodiments of the invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Embodiment of the invention can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a node-readable storage device or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

Method steps of embodiments of the invention can be performed by one or more programmable processors executing a computer program to perform functions of the invention by operating on input data and generating output. Method steps can also be performed by, and apparatus of the invention can be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or

transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in special purpose logic circuitry.

To provide for interaction with a user, embodiments of the invention can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

Embodiments of the invention can be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of embodiments of the invention, or any combination of such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Other embodiments are within the scope of the following claims.